Detecting and Pruning Introns for Faster Decision Tree Evolution

Jeroen Eggermont
jeggermo@liacs.nl

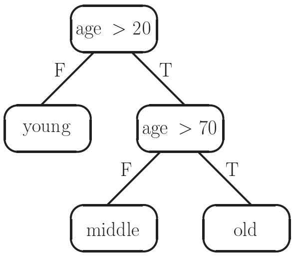
Joost N. Kok
joost@liacs.nl

Walter A. Kosters
kosters@liacs.nl

Universiteit Leiden

Theory

Genetic Programming (GP) can be used for classification. The individuals in the population are decision trees in C4.5 style. Each one tries to handle a given training set in the best way. The trees are manipulated by means of the usual genetic operators (selection, mutation, crossover) in order to find trees with a classification error (on a test set) as low as possible.
An example tree, with F (false) for left children and T (True) for right children:



In the simple GP the internal nodes can have any possible *attribute–operator–value* combination, in the cluster GP unsupervised K-means clustering is used to partition the domain of each numerical valued attribute into a fixed number of clusters.

The decision trees may contain two types of so-called introns:

- subtrees that are never traversed;
- internal nodes that have no influence on the outcome.

Both these types of introns can be pruned during evaluation. First one detects (and removes) intron subtrees in a top-down manner, and then intron nodes are detected (and removed) in a bottom-up way.

Parallel to this, a fitness cache is used to store trees (and their fitness) that have already been evaluated earlier. The fact that semantically equivalent (but syntactically different) trees can be detected, allows for a smaller cache.

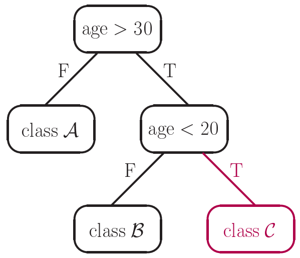
So the goal is to get a faster algorithm, which manipulates and produces smaller trees — that are easier to understand, and of high quality.

Questions

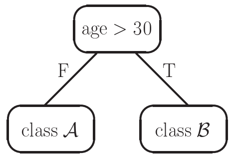
- When is the right moment to prune?
- Does the order of the pruning steps matter?
- How about fuzzy trees?
- Bloat might be beneficial (like in DNA)?

Examples

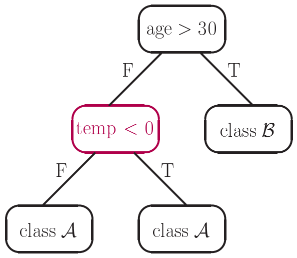
The internal nodes of the trees contain yes/no questions, the leaves contain the classification.



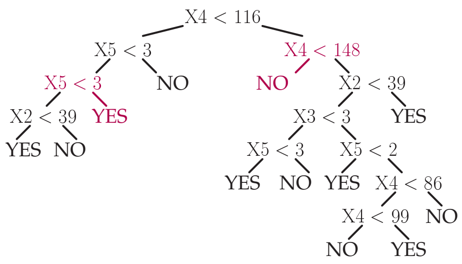
The tree above has an intron subtree that can never be reached — so it can be removed, leading to the semantically equivalent tree



The following tree has an intron node that has no influence on the outcome — so it can be removed, leading to the same tree as in the previous example, again semantically equivalent.



Example of a "best" tree for the German Credit Data set (1,000 records, 23+1 attributes, like age, job and account, and target solvency), using attributes X2, X3, X4 and X5. The number of nodes is reduced from 23 to 19:



Practice

Experiments show that there indeed is a speedup, of course depending on the database at hand. Intron detection and pruning may lead to a 20% to 50% improvement. The simple GP seems to profit the most, but results for the cluster GP are similar.

The cache naturally has a beneficial influence on the number of fitness evaluations. The speedup is somewhat limited due to the overhead, and seems to be heavily dependent on the implementation — and again, on the database at hand.

Experiments on the German Credit Data set (simple GP; results for 100 runs, i.e., 20,000 individuals in total):

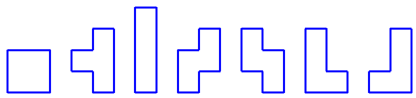
intron detection scheme	pruned tree size					
	best tree			average tree		
	avg	min	max	"avg"	min	max
none	50.0	23	63	36.1	29.8	44.3
nodes	46.4	23	63	30.9	25.3	37.9
subtrees	37.8	19	51	22.2	16.1	32.4
both	36.3	19	47	19.7	14.2	28.6

intron detection scheme	% cache hits			relative runtime
	avg	min	max	
none	15.4	9.5	24.7	1.0
nodes	19.0	11.8	29.2	0.9
subtrees	44.1	25.4	56.9	0.7
both	51.1	30.1	65.2	0.6

And for the cluster GP:

intron detection scheme	pruned tree size					
	best tree			average tree		
	avg	min	max	"avg"	min	max
none	53.1	35	63	36.3	29.9	44.3
nodes	49.1	35	61	30.8	24.7	37.4
subtrees	44.4	31	59	27.0	19.1	38.2
both	41.9	31	53	23.4	17.2	32.5

Note that the reduction in tree size is more spectacular *during* runs than for final trees. Pruning of intron subtrees gives a significant improvement.



Some other research interests of W.A. Kosters:

- Tetris: NP-completeness, decidability
- Data mining: association rules
- Neural networks: Multi Dimensional Scaling

`http://www.liacs.nl/home/kosters/`